Projeto e Implementação de Decodificadores de Código em Plataformas FPGA

Felipe de Oliveira de Araújo[1], **Ricardo Ribeiro dos Santos**[2]

[1]Faculdade de Engenharia (FAENG)
Universidade Federal do Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

[2]Faculdade de Computação (FACOM)
Universidade Federal do Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

felipe.oli_@hotmail.com, ricardo@facom.ufms.br

Abstract. *This paper presents the design, implementation, and characterization of instruction decoders on an ILP (Instruction-Level Parallelism) processor named ρ-VEX. The design and implementation of the instruction decoder is based on the Pattern Based Instruction Word (PBIW) encoding technique. PBIW is an instruction encoding technique independent of the instruction set. The design and implementation of the PBIW instruction decoder enable to evaluate the impacts of the decoding technique on the processor hardware. Specifically, the experiments and results show alternatives for minimizing the power consumption and area of the processor.*

Resumo. *Esse artigo apresenta o projeto, implementação e caracterização de decodificadores de instruções sobre um processador que explora paralelismo em nível de instrução. O hardware projetado e implementado neste trabalho decodifica instruções previamente codificadas usando a técnica PBIW (Pattern Based Instruction Word). PBIW é uma técnica de codificação de instruções que não é dependente de um conjunto específico de instruções. O projeto e a implementação do decodificador de instruções PBIW possibilita avaliar os impactos reais da adoção de uma técnica de decodificação sobre o hardware do processador. Especificamente, os experimentos e resultados obtidos com o hardware decodificador revelam alternativas para minimizar o consumo de potência e a área ocupada pelo processador.*

1. Introdução

A evolução da tecnologia em torno de novos modelos arquiteturais tem chegado até o projeto de processadores embarcados. Atualmente, processadores embarcados utilizam *cores* de processamento e *caches on-chip* para aumentar o desempenho de programas. Diante dessa maior complexidade dos sistemas, torna-se ainda importante considerar questões como redução da área de memória ocupada pelos programas e, principalmente, impactos dessa redução no projeto do sistema computacional.

Do objetivo inicial de reduzir o tamanho do código do programa para questões envolvendo o impacto do descompressor/decodificador no sistema computacional, as

técnicas de compressão/codificação de código focaram no compromisso entre área do programa versus desempenho do código, projeto de baixo consumo de potência e aumento do tempo de ciclo do processador. Assim, embora a preocupação inicial tenha sido a eficiência na compressão/codificação do código, os impactos gerados pela inclusão de um descompressor/decodificador no processador não podem ser desconsiderados.

Este artigo realizou o projeto e a implementação de um circuito decodificador de instruções baseado na técnica de codificação PBIW [Santos et al. 2009] sobre a via de dados do processador ρ-VEX. Especificamente, este trabalho projetou, implementou e avaliou um circuito decodificador de instruções para um projeto de codificação de instruções, sobre o processador ρ-VEX , proposto em [Marks 2012]. A concepção da técnica de codificação de instruções PBIW assim como o projeto e implementação do codificador foi objeto de estudo de um trabalho de mestrado [Marks 2012]. A fim de avaliar os impactos da inserção de um novo circuito junto ao projeto microarquitetural do processador, vários experimentos foram realizados considerando a área ocupada pelo circuito, consumo de potência dissipada e frequência máxima alcançável pelo projeto. O projeto do circuito foi descrito em VHDL (*Very High Speed Integrated Circuits Hardware Description Language*) e implementado sobre uma plataforma de hardware com tecnologia FPGA (*Field Programmable Gate Array*). A escolha da tecnologia FPGA como plataforma para implementação e síntese em hardware se dá pela flexibilidade dessa tecnologia (pode-se programá-la e reprogramá-la) e sua disponibilidade para utilização imediata no Laboratório de Sistemas Computacionais de Alto Desempenho (LSCAD) da UFMS.

O artigo é organizado conforme segue: A Seção 3 descreve as principais características do processador ρ-VEX. A Seção 2 apresenta a técnica PBIW. A Seção 4 apresenta e discute a implementação da técnica PBIW sobre o processador ρ-VEX. A Seção 5 mostra os experimentos realizados e os resultados com a técnica PBIW. A Seção 6 apresenta as principais conclusões e os resultados do trabalho.

2. A Técnica de Codificação de Instruções PBIW

A técnica de codificação *Pattern Based Instruction Word* (PBIW) [Santos et al. 2009] percorre o programa realizando a fatoração de operandos redundantes e opcodes em dois conjuntos: instruções codificadas e padrões. PBIW explora a sobrejeção entre o conjunto de instruções e o conjunto de padrões. A técnica é embasada em algoritmos de fatoração de operandos [Araujo et al. 1998, Ernst et al. 1997]. PBIW é uma técnica de codificação de instruções, uma vez que comprime as instruções mantendo a semântica dos dados na nova instrução codificada.

Uma instrução codificada PBIW é uma estrutura de dados que não contém dados redundantes (registradores ou imediatos) e é armazenada em uma *Cache* de Instruções. Um padrão PBIW é uma estrutura de dados que contém as informações necessárias para reconstituir, junto a um decodificador de instruções, a instrução original para o estágio de execução. O padrão contém ponteiros para as posições da instrução codificada e é armazenado em uma *cache* ou tabela chamada de Tabela de Padrões (P-Tabela ou P-*Cache*). Dessa forma, o decodificador de instruções deve ser implementado junto à via de dados e controle de um processador e, especificamente, antes do estágio/ciclo de execução da instrução. o projeto do decodificador PBIW explora o paralelismo entre as atividades de *hardware*, tornando mais simples e eficiente a decodificação dos campos das instruções

para a unidade de controle e para leitura de operandos no banco de registradores.

Diferente de técnicas de codificação voltadas para conjuntos de instruções específicos [Holdings 2007, Kissell 1997, Ecco et al. 2009], PBIW não é uma técnica de codificação com enfoque em um conjunto de instruções (ISA - *Instruction Set Architecture*) ou processador específico. Assim, o projeto do padrão e instrução codificada deve considerar alguns parâmetros:

- número de registradores de leitura e escrita;
- número e tamanho de imediatos na instrução codificada;
- tamanaho da tabela de padrões.

Cada instrução codificada PBIW tem $\mathcal{R}$ ($0 < \mathcal{R} \leq \mathcal{P}$) campos para representar registradores de leitura, onde $\mathcal{P}$ é o número máximo de portas de leitura do banco de registradores. Cada campo registrador na instrução deve comportar até $\lceil \log_2 \textit{Regs} \rceil$ bits para endereçar cada registrador, onde *Regs* é o número total de registradores da arquitetura. A instrução codificada aponta para seu respectivo padrão através do campo ponteiro do padrão, cujo tamanho é $\lceil \log_2 |\text{P-Tabela}| \rceil$ bits, em que, |P-Tabela| indica a quantidade máxima de padrões que a tabela de padrões pode armazenar. A instrução codificada PBIW pode oferecer campos de cancelamento que são utilizados para anular operações/sílabas específicas da instrução. A quantidade N de bits de cancelamento corresponde ao número de operações que constituem o padrão PBIW.

Na estrutura de dados do padrão está(ão) armazenada(s) a(s) operação(ões) da instrução codificada. Cada operação contém um opcode e um número fixo de operandos. Cada operando tem $\lceil \log_2 X \rceil$ bits, onde X é o número de campos da instrução codificada PBIW dedicados a armazenar os endereços dos operandos da instrução original (escrita ou leitura).

Os campos de cancelamento na instrução codificada PBIW indicam que operações do padrão não serão executadas em tempo de execução. A utilização de campos de cancelamento permite que o padrão seja utilizado por mais de uma instrução codificada (reúso de padrões).

Ao aplicar a técnica PBIW sobre um conjunto de instruções específico, o algoritmo de codificação PBIW acrescentará um novo passo na infraestrutura de complicação para geração de código para o processador alvo. A infraestrutura de codificação de instruções que mapeia o código de saída de um compilador para o esquema de codificação PBIW foi desenvolvido por [Marks 2012]. A complexidade do algoritmo de codificação PBIW é diretamente relacionada ao algoritmo de busca utilizado para verificar se existe um padrão similar ao recém codificado. Caso seja usada uma busca linear sobre o conjunto de padrões, a complexidade assintótica do algoritmo de codificação será $(O(N^2),$ onde $N =$ número de padrões criados$)$ [Marks 2012].

3. O Processador ρ-VEX

O ρ-VEX [van As et al. 2008] é um processador VLIW *soft-core* configurável descrito em VHDL baseado no ISA VEX [Fisher et al. 2005]. O ISA VEX é inspirado no ISA da família de processadores HP/ST Lx [Fisher et al. 2005] (o mesmo usado para os processadores ST200). O ρ-VEX possui uma micro-arquitetura VLIW minimalista: não possui unidade de gerenciamento de memória nem unidade para cálculo de números de ponto-flutuante. Apesar dessas limitações, o processador ρ-VEX possui diversas características,

como instruções longas em memória e projeto *soft-core* embarcado, que o torna adequado a adotar uma técnica de codificação de instruções.

A Figura 1 apresenta a via de dados do processador ρ-VEX. Depois dos estágios de busca e decodificação, as sílabas de uma instrução são despachadas para unidades de execução individuais. Cada instrução contém 4 sílabas. Todas as sílabas podem ser operações lógicas ou aritméticas (A), porém operações de controle (CTRL) só estão presentes na primeira sílaba. Operações de multiplicação e divisão (M) podem estar presentes somente nas segunda e terceiras sílabas e operações de acesso à memória (MEM) só estão presentes na quarta sílaba.

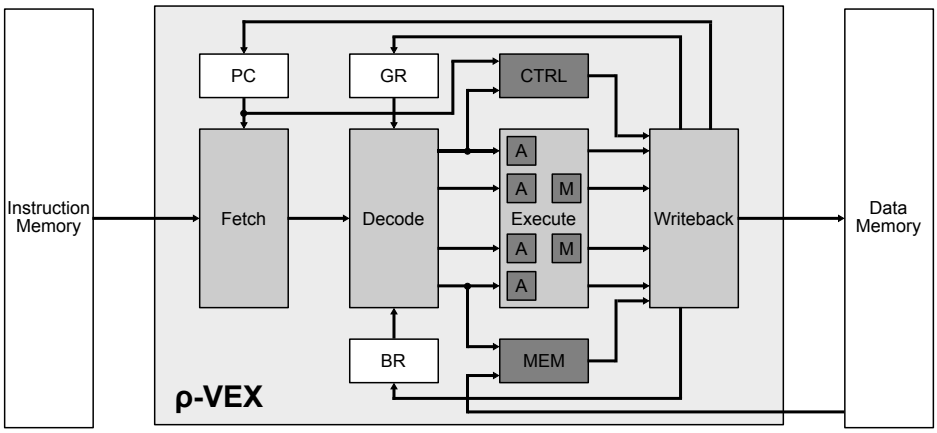

Figura 1. Via de dados arquitetural do processador ρ-VEX.

O conjunto de ferramentas de geração de código e simulação para ρ-VEX contém um compilador C, bibliotecas ANSI C, um simulador em *software* (disponibilizado pela HP [Hewlett-Packard Laboratories 2012]) e o montador ρ-ASM. Apesar da existência de ferramentas para geração e simulação de código, tais ferramentas não suportam a compilação e simulação de qualquer tipo de programa. Especificamente, não há ferramentas de link-edição, forçando, assim, a remoção de todas as chamadas a funções presentes em bibliotecas externas. Todas as funções em um programa devem ser expandidas *inline* na função *main* de forma a preservar o endereçamento global de variáveis. A conversão para código inline de funções é um passo obrigatório pois o *toolchain* ρ-VEX não suporta manipulação de pilha.

4. Projeto da Codificação e Decodificação PBIW sobre o Processador ρ-VEX

De acordo com os parâmetros a serem considerados para realizar a codificação PBIW, projetou-se uma codificação sobre o processador ρ-VEX cujo formato é apresentado na Figura 2. Nesse projeto de codificação PBIW, objetivou-se reduzir o tamanho da instrução ρ-VEX ao máximo possível, considerando a possibilidade de utilizar diferentes tipos de operandos na instrução, reusar operandos e reusar padrões para diferentes instruções (uso dos bits de cancelamento). A instrução ρ-VEX original possui tamanho de 128 bits enquanto que a nova instrução codificada possui 64 bits. Cada instrução codificada possui um campo de ponteiro para o respectivo padrão PBIW (7 bits), o campo de cancelamento (4 bits) e 11 campos indexados de 1 a 11. Esses 11 campos podem ser indexados tanto para operandos de leitura quanto de escrita do Banco de Registradores GR. Os campos

indexados de 1 a 7 são utilizados para armazenar os operandos de leitura e escrita do Banco de Registradores BR. Os campos de operandos possuem 5 bits cada, exceto o último campo, que possui 3 bits. Os últimos 3 campos são usados para armazenar valores imediatos.

O padrão PBIW (Figura 3) possui 96 bits divididos em 4 sílabas de 24 bits cada. Uma sílaba contém um campo de opcode de 7 bits, um campo de 2 bits para seletor de imediato, três campos de 4 bits e um campo de 3 bits usados como índices para os operandos na instrução PBIW.

No conjunto de instruções VEX adotado pelo processador ρ-VEX , operações de desvio como BR e BRF, que possuem seus últimos três bits do opcode fixo, devem armazenar seus endereços de operando de leitura de *branch* nos campos 5 e 6, respectivamente, da instrução PBIW.

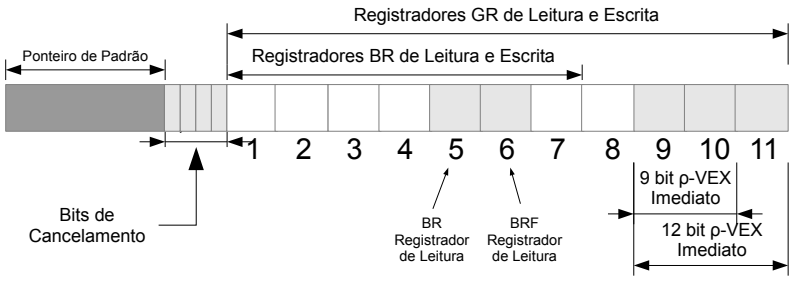

Figura 2. Instrução codificada PBIW para o processador ρ-VEX.

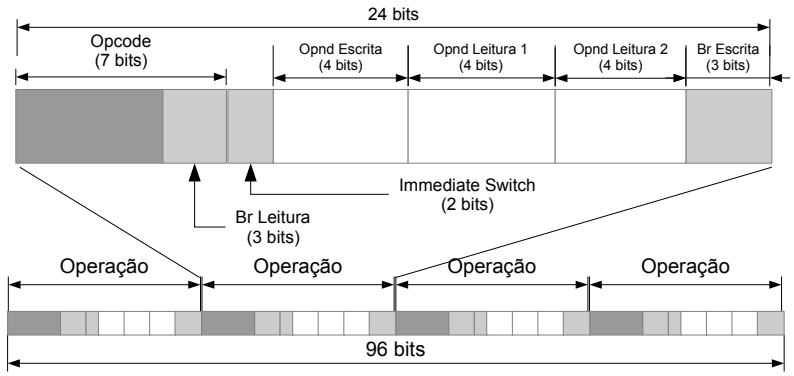

Figura 3. Padrão PBIW para o processador ρ-VEX.

O processo de decodificação PBIW é composto pelos seguintes passos:

- Realiza-se a busca da instrução codificada na memória;
- No estágio de decodificação, o padrão é recuperado na cache de padrões utilizando o índice armazenado na instrução PBIW;
- Os campos da instrução PBIW são disponibilizados nas entradas dos multiplexadores do circuito decodificador (Figura 4);
- Após a recuperação do padrão na cache de padrões os campos que contêm os endereços dos operandos de origem (leitura) e os operandos de destino (escrita) são selecionados utilizando os ponteiros existentes no padrão.

A visão geral do circuito decodificador PBIW é apresentada na Figura 4. Ressalta-se que a Figura 4 detalha o esquema de decodificação para uma operação da instrução ρ-VEX. Entretanto, o mesmo processamento (em paralelo) é realizado para a decodificação das demais operações existentes na instrução.

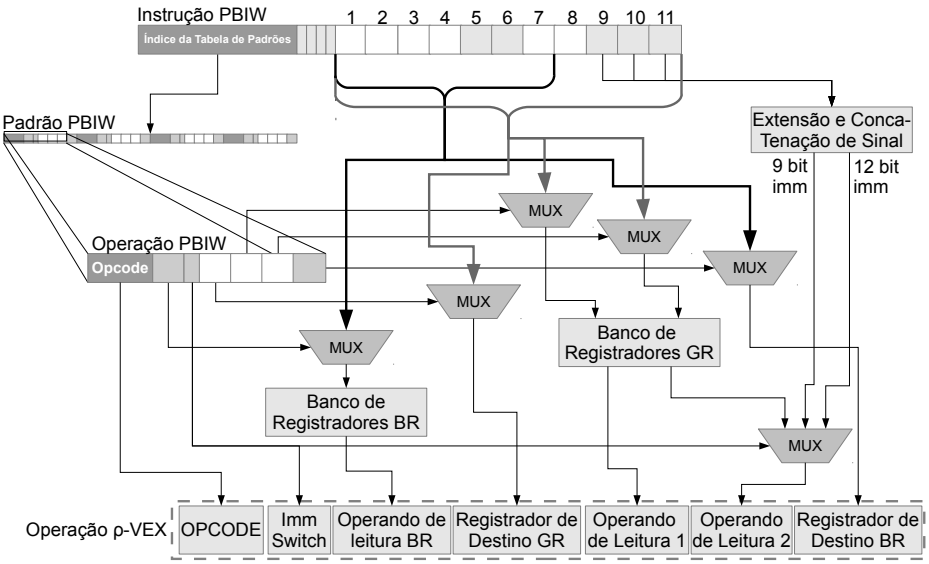

Figura 4. Diagrama de blocos do decodificador PBIW para o processador ρ-VEX.

A implementação da cache de padrões (P-cache) para o decodificador PBIW tem um tamanho de 1,5 Kbytes (128 linhas $\times$ 96 bits = 12.288 kbits= 1,5 Kbytes) e está implementada como uma tabela de mapeamento direto junto a unidade *Decode* do processador ρ-VEX.

5. Experimentos e Resultados

Esta seção apresenta os resultados obtidos com a utilização da técnica PBIW sobre o processador ρ-VEX. Discute-se, em particular, o efeito do circuito decodificador junto à via de dados do processador com relação à área, consumo de potência dinâmica e frequência máxima de operação do circuito implementado sob a execução de um conjunto de 13 programas de propósito geral.

A Tabela 1 apresenta os programas e as instâncias de entrada adotadas nos experimentos. Todos os 13 programas foram escritos em linguagem C. Alguns (5 programas) fazem parte do Trimaran Compiler Suite (*Simple benchmarks*) [Chakrapani et al. 2004]. Os programas *mergesort*, *counting_sort*, *linked_list*, *doubled_linked_list*, *avl_tree*, *kruskal*, *prim* e *floyd_warshal* são implementações de algoritmos clássicos de ordenação e manipulação de estruturas de dados. Todos os programas foram compilados e simulados usando o toolchain VEX [Fisher et al. 2005].

Nos experimentos com o decodificador PBIW, a tecnologia alvo utilizada para prototipar o processador ρ-VEX com o circuito decodificador foi um kit FPGA Altera DE2 da família Cyclone II (EP2C35F672C6). Como ρ-VEX não possui hierarquia de memória, todos os programas foram inseridos na memória de instruções do *soft-core* do ρ-VEX antes da síntese do projeto. As memórias de instrução e dados foram prototipadas utilizando os blocos de memória embarcadas denominados M4K [ALTERA 2008]. Os programas

foram inseridos nas memórias embarcadas utilizando o arquivo "MIF"(*Memory Initialization File*). Os blocos M4K são capazes de implementar vários tipos de memória, por exemplo: *single-port RAM*, *ROM*, *FIFO buffers* entre outros [ALTERA 2008].

Programas	**Descrição**	**Entrada**
avl_tree	Balan. de Árvores	Inserção: 1 a 12 inteiros
nested_complex	Encadeamento de loops	Default
mergesort	Ordenação	Vetor: 15 inteiros
counting_sort	Ordenação	Vetor: 20 inteiros
linked_list	Estrutura de dados	Inserção: 28 inteiros
double_linked_list	Estrutura de dados	Inserção: 17 inteiros
kruskal	Árvore geradora mín.	5x5 matriz de adj.
strcpy	Cópia de strings	strings de 55 bytes
prim	Árvore geradora mín.	3x3 matriz de adj.
bmm	Mult. de Matrizes	vars: NUM=5, BLOCK=1
floyd_warshall	Algo. Caminho mínimo	5x5 matriz de adj.
sqrt	Método Newton Raphson	var: NUM=40
hyper	Desvios internos à loop	200 iterações

Tabela 1. Programas usados nos experimentos.

Os experimentos de caracterização de dissipação de potência e frequência máxima foram realizados utilizando as ferramentas PowerPlay Power Analyzer e o TimeQuest Timing Analyzer integradas à ferramenta Altera Quartus^{TM} [ALTERA 2012a, ALTERA 2012b]. A taxa de comutação, ou taxa de atividade, para o experimento de dissipação de potência foi fornecida para a ferramenta através de um arquivo de simulação (*.vcd) construído a partir do *testbench* da execução de cada um dos 13 programas.

5.0.1. Área e Dissipação de Potência

A Figura 5 mostra o número médio de elementos lógicos das unidades existentes na via de dados do processador com a inserção do decodificador. A inserção do circuito decodificador trouxe um aumento na área da unidade de decodificação de 228% (média entre os 13 programas utilizados) quando comparado com a versão original do ρ-VEX. Esse aumento pode ser explicado pelo fato da tabela de padrões e a lógica do circuito decodificador (multiplexadores) estarem instanciadas na unidade *Decode*.

A adoção da técnica PBIW acarretou maior ocupação de operações em uma instrução. Essa situação gerou um aumento de 8% na área ocupada pelo banco de registradores GR quando comparada com a versão original do processador. Esse aumento pode ser explicado pela mudança no mecanismo de leitura dos dados no banco de registradores GR. No decodificador PBIW, o endereço armazenado na instrução codificada PBIW deve ser multiplexado primeiramente para depois ser buscado nos Bancos de Registradores.

Na unidade *Execute* houve redução significativa (45%) na área ocupada. Essa redução deve-se à redução de circuitos comparadores e registradores para identificar a funcionalidade de cada campo da instrução. A adoção da codificação PBIW reduz a sobreposição de funcionalidades nos campos da operação uma vez que a instrução codificada possui campos com finalidades fixas.

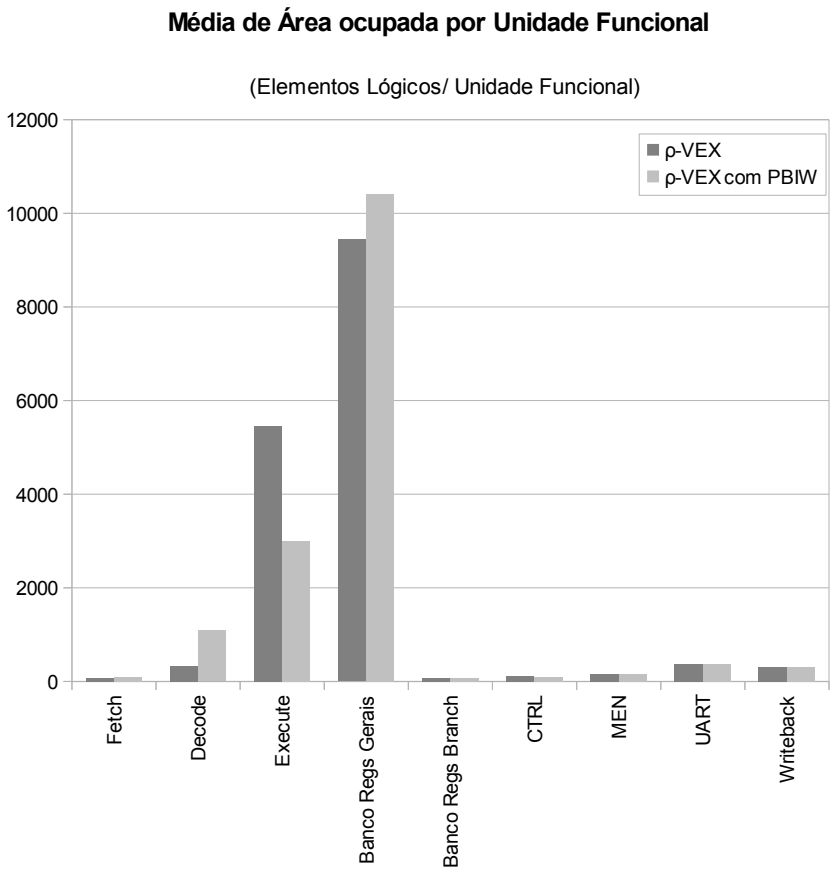

Figura 5. Média de área ocupada por unidade funcional do processador ρ-VEX.

A potência total dissipada por um dispositivo FPGA é constituída por três componentes: Potência Dinâmica, Potência Estática e Potência de Entrada e Saída. A potência de Entrada e Saída é a energia consumida devida à carga e descarga dos condensadores externos ligados aos pinos dos dispositivos externos. A potência dinâmica é a energia consumida para a operação do dispositivo. A potência estática é proporcional ao número de transistores em *off-stage*.

A inserção do decodificador PBIW, junto à microarquitetura do processador ρ-VEX trouxe uma redução média no número de elementos lógicos de 15%. Esse resultado não alterou, de maneira significativa, o consumo de potência estática. Dessa forma, nesse artigo investiga-se apenas os impactos que a técnica de codificação PBIW traz sobre a potência dinâmica.

Embora a unidade *Decode* tenha tido um aumento em sua área, o consumo da potência dinâmica praticamente se manteve constante. Na realização dos experimentos foi possível observar que a dissipação de potência na unidade *Decode* aumenta à medida que há um número maior de campos da operação com a mesma funcionalidade. Além disso, a redução da sobreposição em conjunto com a utilização dos bits de cancelamento reduziu o número de acessos desnecessários a unidades como: CTRL, *Writeback*, *Execute*. Essa eficiência no controle do acesso desnecessário às unidades proporcionou uma redução de potência dissipada de 51%, 53% e 50% nas unidades *CTRL*, *Writeback* e *Execute*, respectivamente.

O reúso dos padrões reduziu a taxa de comutação (*toggle*) de sinal na memória de instrução em 48% em média para o projeto com mecanismo decodificador. Outro efeito da codificação PBIW sobre a memória de instrução foi a redução no número de bits de memória em 62% em média. A consequência dessas reduções tanto na taxa de comutação como no total de número de bits na memória de instrução pode ser vista na Figura 6 que mostra uma redução de 60% no consumo de potência dinâmica na memória de instrução (*imem*). Em média, a inserção do circuito decodificador PBIW junto à microarquitetura do processador ρ-VEX trouxe uma redução de 30% no consumo de potência dinâmica do processador.

A frequencia máxima de operação do processador com o circuito decodificador de instruções também foi avaliada e pode-se observar que o circuito decodificador PBIW sem restrição influencia o caminho crítico (atraso máximo entre uma entrada e uma saída no circuito) da via de dados do processador. A origem na redução da frequência máxima com a inserção do circuito decodificador PBIW está no aumento do tempo de chegada de dados na unidade *Execute*. Este atraso acontece porque a codificação PBIW possibilita que um mesmo imediato seja usado por até quatro diferentes operações na instrução, gerando assim mais comparações com o mesmo imediato. Com a codificação PBIW um mesmo imediato pode ser usado por quatro diferentes operações na instrução, ao contrário da instrução VEX que possui campos separados para armazenar imediato de uma determinada operação quando necessário. Aliado ao aumento do número de operações em um padrão, devido à otimização da junção de padrões, o circuito decodificador PBIW aumentou o fluxo de dados nas entradas dos multiplicadores causando, assim, um atraso no tempo de chegada.

O caminho crítico é definido como o caminho entre uma entrada e uma saída com

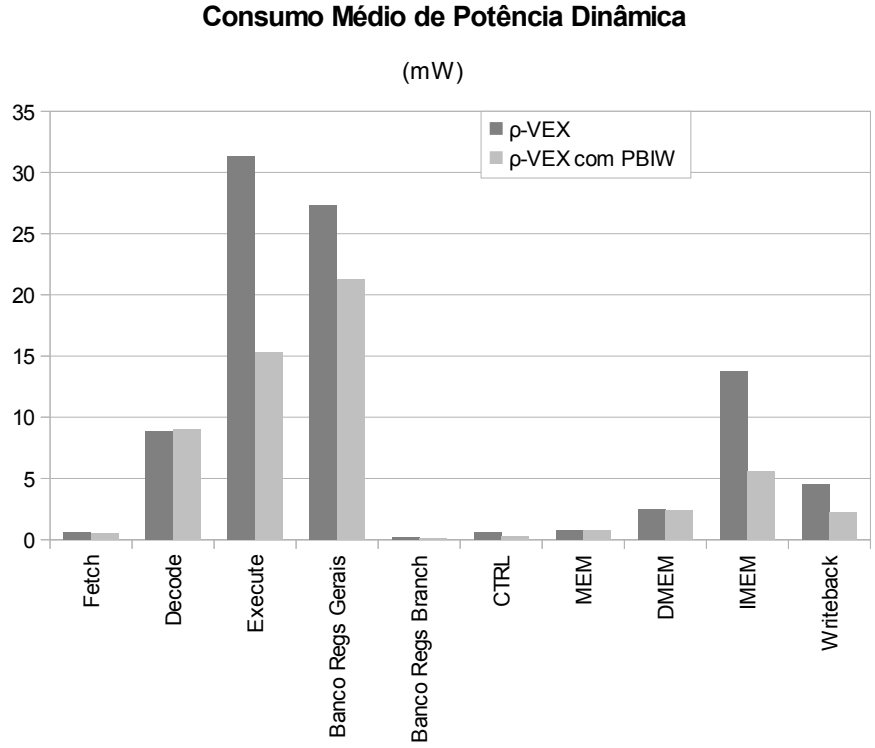

Figura 6. Média de potência dinâmica consumida por unidade funcional.

o atraso máximo. A redução na freqüência de clock máxima é causada pelo aumento do *rising slack* no maior caminho crítico. O *rising slack* é o tempo associado à diferença entre o tempo requerido de um dado e do tempo em que o dado chega, na realidade, a cada conexão. O tempo de chegada (arrival time) de um dado é o tempo decorrido para que um sinal chegue a um determinado ponto. O tempo requerido (required time) é o tempo máximo que um dado pode chegar sem fazer com que o ciclo de relógio seja atrasado.

As Tabelas 2 e 3 apresentam o impacto no tempo de chegada (arrival time), em microsegundos, na execução dos programas avl_tree(pior caso) e counting_sort (melhor caso), respectivamente.

Unidades do processador ρ-VEX	Tempo de chegada ρ-VEX	Tempo de chegada ρ-VEX com decodificador PBIW
Clock Path	2,746	3,811
Instruction Memory	3,202	3,202
Decode	4,889	1,794
Writeback	2,952	0
Registers file GR	5,266	0
Execute	0	13,777
Total tempo de chegada do sinal	19,055	22,087

Tabela 2. Caminho crítico na execução do programa avl_tree

Unidades do processador ρ-VEX	Tempo de chegada ρ-VEX	Tempo de chegada ρ-VEX com decodificador PBIW
Clock Path	2,723	3,549
Instruction Memory	3,202	3,202
Fetch	0	0,595
Decode	1,505	4,418
WriteBack	0	2,343
Register GR	0	5,243
Execute	16,008	0
Total tempo de chegada do sinal	23,438	19,350

Tabela 3. Caminho crítico na execução do programa counting_sort

6. Considerações Finais

Este artigo apresentou a implementação de um decodificador de instruções, baseado na técnica de codificação PBIW aplicada sobre um processador *soft-core* embarcado denominado ρ-VEX. Em particular, apresentou-se os efeitos do mecanismo decodificador PBIW sobre o projeto do processador ρ-VEX. Embora não seja limitada a apenas um tipo de conjunto de instruções, a técnica PBIW mostra-se uma alternativa viável para reduzir a área de utilização do programa na memória e a área utilizada pelo processador. Além disso, a potência dinâmica do processador também pode ser reduzida com a adoção de técnicas de codificação/decodificação de instruções.

Os resultados obtidos com essa técnica demostraram que a codificação PBIW apresenta uma taxa de compressão significativa o que é refletido pela redução na área da memória de instruções. Especificamente, os experimentos em hardware revelam que o mecanismo decodificador PBIW trouxe redução na área total ocupada pelo hardware do processador em 15%. Em média, houve tambem redução de potência dinâmica de 30%. Apesar dos ganhos sobre a redução de área e potência dinâmica com a adoção do decodificador PBIW, pode-se observar uma redução na frequencia máxima de operação do processador de, até, 10%.

Embora não apresentados neste artigo, foram realizados experimentos de desempenho considerando a taxa de acertos na memória de instruções e na *Cache* de padrões. Comparando as taxas de acerto entre os programas sem e com codificação sob a mesma configuração, tamanhos de memórias e controladores de cache, foi comprovado que os programas codificados com a técnica PBIW apresentam uma melhoria significativa de desempenho uma vez que máximiza a taxa de acertos (instruções menores e em maior quantidade na memória) e reduz o tempo de busca de instruções.

Como trabalhos futuros objetiva-se estender a capacidade de manipulação de programas de benchmarks no processador ρ-VEX. Busca-se também comparar detalhadamente o comportamento do processador ρ-VEX com e sem o decodificador PBIW sobre a implementação de ASICs com tecnologia de .35μm. Além de também estender a aplicação da técnica PBIW para processadores escalares.

7. Agradecimentos

Os autores agradecem às agências de financiamento CAPES, CNPq e Fundect-MS pelo apoio financeiro nos projetos de pesquisa desenvolvidos no Laboratório de Sistemas Computacionais de Alto Desempenho (LSCAD/FACOM/UFMS) assim como bolsas de estudo atribuídas aos estudantes de graduação e pós-graduação desse laboratório.

Referências

ALTERA (2008). *Cyclone II Memory Blocks*. Altera Corporation.

ALTERA (2012a). *PowerPlay Power Analysis*. Altera Corporation.

ALTERA (2012b). *The Quartus II TimeQuest Timing Analyzer*. Altera Corporation.

Araujo, G., Centoducatte, P., Cortes, M., and Pannain, R. (1998). Code Compression Based on Operand Factorization. In *Proceedings of the MICRO*, pages 194–201. IEEE Computer Society.

Chakrapani, L. N., Gyllenhaal, J., Mei, W., Hwu, W., Mahlke, S. A., Palem, K. V., and Rabbah, R. M. (2004). Trimaran - An Infrastructure for Research in Instruction-Level Parallelism. *Lecture Notes in Computer Science*, 3602:32–41.

Ecco, L. L., Lopes, B. C., Xavier, E. C., Pannain, R., Centoducatte, P., and de Azevedo, R. J. (2009). Sparc16: A new compression approach for the sparc architecture. In *Proceedings of the SBAC-PAD*, pages 169–176, Washington, DC, USA. IEEE Computer Society.

Ernst, J., Evans, W., Fraser, C. W., Lucco, S., and Proebsting, T. A. (1997). Code compression. In *Proceedings of the PLDI*, pages 358–365. ACM.

Fisher, J. A., Faraboschi, P., and Young, C. (2005). *Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools*. Elsevier.

Hewlett-Packard Laboratories (2012). VEX Toolchain. http://www.hpl.hp.com/downloads/vex/.

Holdings, A. (2005-2007). ARM1156T2-S Technical Reference Manual. Technical report. pp. 35-36.

Kissell, K. D. (1997). Mips16: High-density MIPS for the embedded market. Real Time Systems.

Marks, R. A. (2012). Infraestrutura para Codicação de Instruções Baseada em Fatoração de Padrões. Master's thesis, Universidade Federal do Mato Grosso do Sul.

Santos, R., Batistella, R., and Azevedo, R. (2009). A pattern based instruction encoding technique for high performance architectures. *International Journal on High Performance Systems and Architecture*, 2:71–80.

van As, T., Wong, S., and Brown, G. (2008). ρ-VEX: A Reconfigurable and Extensible VLIW Processor. In *Proceedings of the FPT*. IEEE.